Milestone 2 Progress Evaluation

Project Information

Project Title :
Skin Cancer Detection App Using Machine Learning

Names and Email Addresses :
Samuel Borges - borgess2023@my.fit.edu
Elif Dila Bayrak - ebayrak2025@my.fit.edu
Neva Ildem - zildem2023@my.fit.edu
Sophia Zubieta - szubieta2023@my.fit.edu

Faculty Advisor :
Zahra Nematzadeh - znematzadeh@fit.edu

Client :
Dr. Zahra Nematzadeh, Department of Electrical Engineering and Computer Science, FIT

Progress of the current Milestone

Milestone 2 Progress Matrix :

Task	Completion %	Elif	Neva	Samuel	Sophia
Dataset & Preprocessing Pipeline Execution	100%	0%	50%	0%	50%
CNN Model Training	100%	50%	50%	0%	0%
Mobile App Frontend UI	100%	0%	0%	100%	0%
REST API Backend & Preprocessing Endpoint	100%	0%	0%	0%	100%
Model Evaluation & Confusion Matrix	100%	50%	50%	0%	0%
App to Server Image Upload	100%	0%	0%	50%	50%

Discussion of Accomplished Tasks and Obstacles for the Current Milestone

Task 1: Dataset & Preprocessing Pipeline Execution :

Dataset preparation was done using the ISIC 2019 dataset; metadata and ground-truth labels were loaded. Along with auditing class distributions, removing corrupt image files, duplicate images, and resolving conflicting labels. Preliminary benign and malignant classifications were established, excluding Actinic Keratosis, and investigating missing lesion IDs. This process involved examining the original nine diagnostic classes and deciding how to map them to binary labels for model training. Special attention was given to ambiguous classifications and inconsistent metadata to ensure that the resulting dataset would be appropriate for a reliable model evaluation. Stratified train, validation, and test splits were also created. These splits were created while also ensuring images from the same lesion ID were grouped together without split overlap. This was particularly important because multiple images could correspond to the same lesion, which could otherwise introduce data leakage and produce misleading evaluation results. The resulting datasets were verified to ensure that no image IDs or lesion IDs overlapped between the training, validation, and test sets. A cleaned binary dataset was uploaded to Google Drive, with an initial training subset of 1003 images for model training. The smaller training subset was used to make initial model experimentation more manageable with the available computational resources. Following preprocessing and image quality filtering, 893 training images and 2,997 validation images were available for training and evaluating the CNN and transfer learning models. Then the preprocessing pipeline was designed and executed, including aspect-ratio padding to prevent distortion, image resizing to 224x224, noise reduction, and tensor normalization. During this process, we discovered an unbalanced class distribution and duplicate images; we were able to resolve these problems using deduplication, lesion-level grouping, and stratified splitting. On the preprocessing side, in order to fix the resizing distortion, black padding was implemented, but following discussion, it was decided to change this padding to image reflection and blur padding. In addition, image blur was affecting model accuracy, and a quality gate was established to reject unreadable photos before preprocessing. The dataset preparation and preprocessing stages were coordinated to ensure that the resulting images and binary labels were consistent across the different machine learning architectures. The independent test dataset was kept separate from model training and validation to prevent test data from influencing model selection and hyperparameter optimization. The independent test dataset was also processed using the same preprocessing pipeline applied to the training and validation datasets. Of the original 3,232 test images, 2,823 passed the image quality filtering process, while 409 were rejected. The accepted test dataset contained 1,843 benign images and 980 malignant images, all preprocessed to 224 x 224 pixels. The processed images and corresponding CSV file were transferred to AI.Panther, where image availability, label alignment, and dataset integrity were verified. This independent test dataset was reserved for the final model comparison and was not used during model training, hyperparameter tuning, or threshold selection.

Task 2a: CNN Model Training :

We developed and trained a basic Convolutional Neural Network (CNN) using the ISIC2019 dataset to classify skin lesions as benign or malignant. We used 893 preprocessed training images and 2,997 validation images. Two versions were trained for 5 and 10 epochs. The 10-epoch model achieved 69.60% validation accuracy using the default classification threshold. Both trained models were saved to our shared Google Drive for future testing and comparison.

Task 2b: Transfer Learning Models Training :

To categorize skin lesion images as benign or malignant, transfer learning models were created and trained using the preprocessed ISIC 2019 dataset. MobileNetV2 and EfficientNetV2-B2, two pretrained architectures, were chosen for additional testing and comparison. Python and PyTorch were used to train both models on the AI.Panther GPU environment, using 2,997 validation images and 893 preprocessed training images. To examine various training configurations and enhance classification performance, multiple experiments were conducted for both architectures. Learning rates, weight decay, the number of unfrozen layers, and other training techniques, including data augmentation and label smoothing, were also changed during these tests. Throughout the tests, model checkpoints were saved, enabling the selection of the top-performing configurations based on validation outcomes.

Configuration 12 from Experiment 4 was chosen as the best-performing MobileNetV2 model, with its checkpoint saved at epoch 6 after a grid search examined multiple hyperparameter combinations. Experiment 4, which included label smoothing, was chosen for EfficientNetV2-B2, and its best checkpoint was stored at epoch 25. Optimized classification thresholds derived from validation data were used to further assess both models. For MobileNetV2 and EfficientNetV2-B2, the chosen thresholds were 0.40 and 0.30, respectively. Through testing and model configuration modifications, issues with insufficient training data, class imbalance, overfitting, and differences in classification performance were investigated and addressed through adjustments to the training configurations. For further evaluation and application integration, the trained models and their chosen configurations were stored.

Task 3: Mobile App Frontend UI :

Continuing the development of the user interface of the mobile application, we extended the navigation system to open a new screen by pressing the “Capture/Upload Image” placeholder. This new interface provides users with a "Take Photo" button that opens the device's camera, allowing them to capture an image directly from the application. After taking a photograph using the device's camera, the user is given the option to retake or confirm the captured image. Once the photograph is confirmed, the user can upload it to the Flask API server for preprocessing. After the image has been successfully uploaded, the backend performs image preprocessing and quality verification. Once preprocessing is complete, the application displays a blur score and indicates whether the image passed the quality verification. This functionality was implemented

to test the integration between the frontend and backend program and evaluate the quality of the uploaded images. Additionally, the interface provides feedback if the upload or connection is unsuccessful, which may occur due to network connectivity issues or problems communicating with the Flask server.

Task 4: REST API Backend & Preprocessing Endpoint :

Developed and established the first backend construction of the project. A Flask REST API was previously decided upon, and implementation was begun. First, a backend repository was created using GitHub. The goal was for this API backend to eventually have a server connection stemming from the client's application on their mobile device to the model the group has established to complete the full pipeline. This implementation was begun by establishing a /health monitoring endpoint along with a /predict endpoint that together handle the image uploads to the server. In addition to this, a standardized JSON payload contract was created with the ability to return a binary label of either benign or malignant. This is a placeholder for the final product. As of Milestone 2, analysis of models is still underway, and a concrete model has not been fully chosen and developed, as a result, a mock inference model has been put in its place, this allows for continued testing from the front-end API connection testing, while model testing is ongoing.

Task 5: Model Evaluation & Confusion Matrix :

We evaluated both CNN models using accuracy, precision, recall, F1 score, and confusion matrices. The 10-epoch model achieved 69.60% validation accuracy. We also tested different classification thresholds (0.30, 0.40, and 0.50) to improve malignant detection. At a threshold of 0.30, malignant recall increased to 85.56%, reducing false negatives from 378 to 146. However, this also increased false positives and reduced overall accuracy to 59.49%. These results helped us understand the trade-off between accuracy and malignant detection.

The same 2,997-image validation dataset was used to evaluate MobileNetV2 and EfficientNetV2-B2, in addition to CNN. With a classification threshold of 0.40, the final MobileNetV2 configuration obtained 72.17% validation accuracy, 81.11% malignant recall, an F1 score of 0.6629, and an ROC-AUC of 0.8277. With a threshold of 0.30, the chosen EfficientNetV2-B2 configuration produced 69.67% validation accuracy, 70.23% malignant recall, an F1 score of 0.6097, and a ROC-AUC of 0.7581. According to confusion matrix analysis, EfficientNetV2-B2 correctly detected 710 malignant cases and missed 301, whereas MobileNetV2 correctly identified 820 malignant instances and missed 191. A paired bootstrap analysis was also performed using 2000 resamples. It was found that MobileNetV2 achieved a mean improvement in ROC-AUC of 0.0697 (95% CI: [0.0526, 0.0859]). This was alongside improvements in the F1 score and malignant recall. The MobileNetV2 model was therefore chosen as the better transfer learning model based on the validation results.

Both transfer learning models performed well after validation-based model selection, with MobileNetV2 achieving 71.87% accuracy on the test dataset (2,823 images), with 80.20% recall for malignant cases, and an F1 score of 0.6644 and ROC-AUC score of 0.8184. The EfficientNetV2-B2 model achieved 66.99% accuracy, 66.12% recall for malignant cases, an F1 score of 0.5817, and an ROC-AUC of 0.7353. Analysis of the confusion matrix indicated that MobileNetV2 correctly identified 786 malignant cases but failed to identify 194 cases. In contrast, the EfficientNetV2-B2 correctly identified only 648 malignant cases but failed to identify 332 cases. A paired bootstrap analysis was conducted for each lesion group using 2000 resamples, indicating improved performance of MobileNetV2 over EfficientNetV2-B2. The mean ROC-AUC score improved by 0.0834 with 95% CI [0.0598, 0.1090]. This confirmed MobileNetV2 as the best transfer learning option to integrate into the application.

Task 6: App to Server Image Upload :

In order to accomplish the app-to-server image upload, we configured the /predict endpoint to accept image payloads. The endpoint was then integrated with the Flutter application using the HTTP package, allowing users to upload images to the server through a form-data request. With the Flask REST API program running, it receives an image, extracts the uploaded file, and passes it through the preprocessing pipeline, which includes image-quality verification, noise reduction, resizing, sharpening, and normalization. After preprocessing is complete, the server returns a JSON response containing information about the image's quality and preprocessing status. The Flutter application then interprets this response and displays the blur score and whether the image passed quality verification. This successfully shows that the Flutter application can establish a connection between the frontend and backend, demonstrating that the image can be transmitted to the server, preprocessed, and evaluated for image quality.

Contribution of Each Member to the Current Milestone

- **Elif Bayrak:** For Milestone 2, Elif worked on developing, training, and evaluating the CNN model for skin cancer classification. I trained two CNN versions using 5 and 10 epochs, compared their performance, and generated confusion matrices and training graphs. I also tested different classification thresholds to improve malignant detection and reduce false negatives. Finally, I saved the trained models and organized the results in our shared Google Drive.
- **Neva Ildem:** For Milestone 2, Neva focused on preparing the dataset, developing a transfer learning model, and evaluating it for skin cancer classification. In developing the ISIC 2019 dataset, she helped examine the diagnostic labels. She resolved any inconsistencies in the metadata and created lesion-level training, validation, and test sets to avoid data leakage. To develop the MobileNetV2 and EfficientNetV2-B2 models, she used PyTorch for AI.Panther to train and optimize the models. This involved running multiple experiments, including hyperparameter tuning, data augmentation, and label

smoothing. The best-performing checkpoints and classification thresholds were chosen based on the validation results. She then evaluated both models using classification metrics, confusion matrices, ROC curves, precision and recall curves, and paired bootstrapping. Finally, she prepared and validated the final test set, which includes 2823 accepted images for future evaluation of the finalized models. The two transfer learning models were tested on the testing data, confusion matrices were created, and ROC and precision-recall curves were plotted. A paired bootstrap analysis was performed to compare the results of the two models using grouped lesions. This confirmed that MobileNetV2 is the preferred choice of transfer learning model. The final checkpoint file was packaged up along with the results for use in future integration.

- **Samuel Borges:** For this milestone, Samuel focused on further developing the user interface of the mobile application and worked on establishing a connection between the frontend and backend. When developing the user interface, he implemented a navigation screen for the Capture/Upload Image button, which will give the user the ability to take a photo using the camera device. After taking the photo, the user has the option to upload the image. Samuel also established a connection between the Flutter application and the Flask REST API server by integrating the HTTP package. He initially tested the connection using the ADB reverse method, which allowed the Samsung phone to communicate with the Flask server running locally on his computer by using a USB cable for transferring data. After successfully establishing this connection, he configured the application to use the laptop's Wi-Fi LAN IP address, allowing the phone to communicate with the server wirelessly. This method was also tested successfully, confirming that images captured through the Flutter application could be transmitted to and received by the Flask server for preprocessing using the campus internet connection.
- **Sophia Zubieta:** For this current milestone, Sophia worked on implementing the Flask REST API backend architecture along with redesigning the image preprocessing system. She began by working on reconfiguring the image preprocessing pipeline. This included changing the blur threshold from 100.0 to 15.0. In addition, aspect-ratio padding was improved upon; instead of using a black bar as padding, the image was reflected and blurred, and this was used as padding instead. She was then able to develop a Flask REST API backend server to act as the bridge between the front-end, which included image upload handling, establishing a JSON API contract with a mock inference to unblock front-end development, and the back-end, which includes the deep-learning model. This included building the /health and /predict endpoints for front-end development. Along with the previously established preprocessing pipeline, including implementing the control gate and configuring the blur detection threshold, aspect-ratio padding, denoising, resizing, and tensor normalization.

Plan for Milestone 3

Milestone 3 Task Matrix

Task	Elif	Neva	Samuel	Sophia
1. Dataset expansion and preprocessing pipeline optimization	0%	50%	0%	50%
2. MobileNetV2 model optimization and performance improvement	0%	100%	0%	0%
3. CNN model optimization and comparative evaluation	100%	0%	0%	0%
4. Start ML model integration into Flask REST API	0%	0%	0%	100%
5. Start getting preliminary mobile application results interface, and prediction display	0%	0%	100%	0%
6. Start user authentication, database, and analysis history	0%	0%	1030%	0%
7. Start end-to-end application integration and functional testing	50%	0%	0%	50%
8. Application reliability, security, and performance testing	50%	0%	50%	0%

Discussion of Planned Tasks for Milestone 3

- **Task 1: Dataset Expansion and Preprocessing Pipeline Optimization:**

  Neva and Sophia will continue to improve the preparation and preprocessing pipeline. This will include the additional cleaning processes that were put on hold, as well as checking for gaps in the dataset and duplicate images. Additionally, further preprocessing will be performed on the training dataset. Experiments will also be conducted to assess image quality, verify consistent preprocessing, and identify potential improvements in the image transformations. The aim is to ensure the dataset can be relied upon, reduce

sources of bias, and ensure it is consistently preprocessed to meet the needs of the machine learning models and the application.

- **Task 2: MobileNetV2 Model Optimization and Performance Improvement:**

  Neva will further develop the chosen MobileNetV2 transfer learning model by exploring improvements in the model performance. Experiments will be conducted to further explore fine-tuning techniques, training methods, data augmentation, and strategies to overcome class imbalance. The improvements in the model will be tested using validation metrics and compared with the Milestone 2 model to assess any performance improvement.

- **Task 3: CNN model optimization and comparative evaluation**

  Elif will also improve the basic CNN model built in Milestone 2 with different training settings and optimization methods. This will involve changing the number of epochs, learning rates, and model architecture to improve the classification accuracy and malignant lesion detection. Further experiments will focus on reducing overfitting and class imbalance through data augmentation and class-weighted loss functions. The optimized CNN model will be validated using validation accuracy, precision, recall, F1 score, and confusion matrices, with particular attention to reducing false negatives for malignant cases. The results will be compared with previous CNN versions and the chosen transfer learning model, MobileNetV2, to evaluate their performance and determine whether the improvements make the basic CNN a competitive alternative.

- **Task 4: Start ML model integration into Flask REST API**

  Sophia will focus on starting the full integration of the trained model into the Flask REST API backend repository. This will involve replacing the current mock model (mock_cnn_predict()) with the real machine learning inference model that has been established. Once the finalized model weights file is established, the server will be able to load the weights into its memory during startup, optimize execution speed, and prevent delays during requests. The preprocessed image tensors will be fed directly into the model's forward pass, and class probabilities will be calculated. This also includes returning binary classification labels and confidence scores within the JSON API contract. In addition implementation of cross-origin resource sharing will be begun to be able to secure asynchronous requests from the front-end application interface during end-to-end testing.

- **Task 5: Start getting preliminary mobile application results interface, and prediction display:**

Samuel will develop a preliminary results interface within the Flutter application to display image analysis information returned by the Flask backend. This will include creating a dedicated results screen capable of presenting image-quality information, predicted classifications, and confidence scores. The interface will be tested using the existing mock CNN predictions, or with the trained CNN model once it becomes available.

- **Task 6: Start user authentication, database, and analysis history**

  Samuel will begin implementing user authentication strategies to improve the existing authentication system, which currently stores user account credentials in database tuples. He will also work on expanding the database to store images of previously captured skin lesions and their associated information. Finally, he will implement navigation functionality for the "View Image History" feature, allowing users to access previously saved images and review relevant information, such as the date and time each photograph was taken.

- **Task 7: Start end-to-end application integration and functional testing**
  Sophia and Elif will begin the execution of end-to-end integration and functional testing using the mobile front-end application and the Flask REST API backend to ensure there is a seamless system interoperability. Here, the goal is to be able to validate and begin the completion of the user workflow. This includes the initial photo capture or upload to the user interface, real-time model inference, along with client-side results rendering. This testing will evaluate key operational scenarios to improve the application.

- **Task 8: Application reliability, security, and performance testing**

  We will evaluate different testing methods to assess the reliability, security, and performance of the application. This will include testing the stability of the Flask server connection, measuring image upload times, and verifying that user accounts, uploaded photographs, and analysis results are protected against unauthorized access.

Faculty Advisor Meetings

Date(s) of meeting(s) with Client during the current milestone:

- Wednesday, October 21st, 2026.
- Monday, October 26th, 2026

Faculty Advisor Feedback on the Current Milestone

- See faculty advisor feedback below.

Faculty Advisor Signature: _____ Date: _____

Evaluation by Faculty Advisor

Faculty Advisor: please detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu

Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Elif	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Neva	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Samuel	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Sophia	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Facutly Advisor Signature: _____ Date: _____